A simple transfer-learning extension of Hyperband

Lazar Valkov*, Rodolphe Jenatton*, Fela Winkelmolen*, Cédric Archambeau⋆
Amazon AWS AI* & Amazon research⋆
`l.valkov@sms.ed.ac.uk {jenatton, winkelmo, cedrica}@amazon.com`

Abstract

Hyperband [1] has become a popular method to tune the hyperparameters (HPs) of expensive machine learning models, whose performance depends on the amount of resources allocated for training. While Hyperband is conceptually simple, combining random search to a successive halving technique to reallocate resources to the most promising HPs, it often outperforms standard Bayesian optimization when solutions with moderate precision are sufficient. In this paper, we propose a model-based extension of Hyperband, replacing the uniform random sampling of HP candidates by an adaptive non-uniform sampling procedure. We show that our extension not only improves the precision resolution of Hyperband but also supports transfer learning, both, within a Hyperband run and across previous HP tuning tasks. We apply the method to the problem of tuning the learning rate when solving linear regression problems and to the optimization of the HPs of XGBoost binary classifiers across different datasets, showing that we favorably compare with recently proposed extensions of Hyperband.

1 Introduction

Applying machine learning (ML) and complex predictive systems requires users to set various knobs and parameter configurations, which are broadly referred to as *hyperparameters* (HPs). While good performance hinges on a careful tuning of those, this process is time-consuming and crucially depends on the level of expertise of the practitioner. ML examples of HPs are the learning rate of stochastic gradient descent algorithms [2] or the architectural choices of deep neural networks [3]. In order to automate the tuning of HPs, various approaches such as random search [4] or Bayesian optimization (BO) [5, 6] have successfully cast this problem as a global black-box optimization problem. In that formulation, querying the black-box function of interest, say f, typically corresponds to a *full* training of the underlying ML model together with its evaluation on some validation data.

Starting from the key observation that training ML models is an *incremental* process controlled by some *resources*, for instance, training set sizes, number of epochs or some amount of time, a recent line of work [7, 8, 9, 1, 10] has proposed to exploit this feature to reallocate resources to the most promising HPs by making cheap and early decisions based on evaluations of f after having consumed few resources. *Hyperband* [1] relies on a successive halving procedure [11], where a pool containing *randomly sampled* HPs is progressively trimmed according to a theoretically-justified resource schedule. Despite its simplicity, Hyperband was proven to outperform BO on many ML-related tuning tasks in regimes where solutions with moderate precision are acceptable. To further improve the precision resolution of Hyperband, several model-based extensions of Hyperband have been recently proposed [12, 13, 14, 10]. All those approaches share the idea of replacing random search (to generate the initial set of HP candidates) by model-based adaptive sampling.

Another line of research consists in accelerating HP optimization by transferring knowledge collected during previous HP tuning tasks. The topic of transfer learning within the context of BO has received much attention over the past few years [15, 7, 16, 17, 18, 19, 20, 21]. In this setting, the scalability of the underlying surrogate model becomes an important aspect as the data coming from many related

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

tuning tasks have to be ingested. In particular, standard Gaussian process (GP) surrogates, whose inference scales cubicly with respect to the number of evaluations [22], become impractical. Possible solutions include using scalable Bayesian neural networks [23, 20] or ensembles of GPs [21]. Having introduced the relevant related work, we next present our contributions:

• We provide a scalable, model-based extension of Hyperband similar to [10]. While [10] uses Tree-of-Parzen-Estimators (TPE) [24] to replace the random sampling of Hyperband, we use the adaptive Bayesian linear regression (ABLR) model developed in [20]. We do so to take advantage of its established scalability and transfer learning capabilities.

• Our method supports *transfer learning* at two levels: within a single run of Hyperband, which is similar to the setting considered by [10], and across different Hyperband runs (on related HP optimization tasks), which was not considered by [10]. Following the philosophy of [10] that emphasizes simplicity as a core feature, we argue that our approach is even simpler, discarding any additional heuristics (e.g., in relation to the control of the exploration).

• We demonstrate the benefits of our methodology on synthetic simulations to tune the learning rates for the stochastic optimization of linear models and on real-world binary classification tasks with XGBoost [25]. Our experimental results indicate that ABLR+Hyperband consistently outperforms the approach proposed by [10].

2 Background

In this section, we recall important background material required to introduce our methodology.

Bayesian optimization (BO). BO is a well-established methodology to optimize expensive black-box functions [6]. It relies on a probabilistic model of an unknown target $f(\mathbf{x})$ one wishes to optimize and which is repeatedly queried until one runs out of budget (e.g., time). Queries consist in evaluations of f at HP configurations $\mathbf{x}^1, \dots, \mathbf{x}^n$ selected according to an explore-exploit trade-off criterion or *acquisition function* [26, 27, 6]. The HP configuration corresponding to the best query is then returned. One popular approach is to impose a GP prior over f and, in light of the observed queries $f(\mathbf{x}^1), \dots, f(\mathbf{x}^n)$, to compute the posterior GP. The GP model maintains posterior mean and variance functions that are required when evaluating the acquisition function for each new query of f. When n is large and scalability matters, the GP surrogate model can be replaced by other models such as random forest [28], sparse GPs [29, 30] or (Bayesian) neural networks [31, 23, 20].

Adaptive Bayesian linear regression (ABLR) [20]. Let us consider T tasks, which consist in the black-box functions $\{f_t(\cdot)\}_{t=1}^T$, one of which we wish to optimize. Those functions are related in some way, e.g., evaluations of a same model on different data sets. Assume $f_t(\cdot)$ is evaluated N_t times, resulting in the data $\mathcal{D}_t = \{(\mathbf{x}_t^n, y_t^n)\}_{n=1}^{N_t}$, also denoted by $\mathbf{X}_t \in \mathbb{R}^{N_t \times P}$ and $\mathbf{y}_t \in \mathbb{R}^{N_t}$ in stacked form. ABLR considers a joint model for the responses $\mathbf{y}_t$ made of two parts. First a shared feature map $\boldsymbol{\phi}_{\mathbf{z}}(\mathbf{x}) : \mathbb{R}^P \mapsto \mathbb{R}^D$ corresponding to a feedforward NN with D output units, where $\mathbf{z}$ collects all its weights and biases. Second, separate Bayesian linear regression surrogates that share the feature map $\boldsymbol{\phi}_{\mathbf{z}}(\mathbf{x})$ to model the black-box functions (noting $\boldsymbol{\Phi}_{\mathbf{z}}(\mathbf{X}_t) = [\boldsymbol{\phi}_{\mathbf{z}}(\mathbf{x}_t^n)]_n \in \mathbb{R}^{N_t \times D}$):

$$P(\mathbf{y}_t|\mathbf{w}_t, \mathbf{z}, \beta_t) = \mathcal{N}(\boldsymbol{\Phi}_{\mathbf{z}}(\mathbf{X}_t)\mathbf{w}_t, \beta_t^{-1}\mathbf{I}_{N_t}), \quad P(\mathbf{w}_t|\alpha_t) = \mathcal{N}(\mathbf{0}, \alpha_t^{-1}\mathbf{I}_D),$$

where $\beta_t > 0$ and $\alpha_t > 0$ are precision (i.e., inverse variance) parameters. The model adapts to the scale and the noise level of the black-box function f_t via β_t and α_t, while the underlying NN parametrized by a shared vector $\mathbf{z}$ learns a representation to transfer information between the black-box functions. The weights $\mathbf{w}_t$ of the Bayesian linear regression models are latent variables that are integrated out, while $\mathbf{z}$ and $\{\alpha_t, \beta_t\}_{t=1}^T$ are learned. ABLR can thus be regarded as a NN whose final linear layers (the $\mathbf{w}_t$'s) are subject to a Bayesian treatment. As further detailed in [20], posterior inference is analytically tractable with an overall computational complexity of $\mathcal{O}(\sum_{t=1}^T \min\{N_t, D\}^2 \max\{N_t, D\})$, while the parameters $\mathbf{z}$ and $\{\alpha_t, \beta_t\}_{t=1}^T$ are jointly learned by optimizing the marginal likelihood with L-BFGS.

Hyperband. The algorithm from [1] addresses the problem of HP tuning by incrementally allocating more resources to the best-performing candidates initially taken from a pool of randomly sampled candidates. Supported by theoretical guarantees for correctness and sample complexity, Hyperband follows a resource-allocation schedule determined from two parameters, namely the fraction $1/\eta$ of candidates to incrementally discard (by default in [1], $\eta = 3$) and the maximum resources to be allocated to a given candidate (e.g., a maximum number of epochs or a certain maximum time budget). This schedule is structured in several independent *brackets* that represent different resource-allocation trade-offs, for instance, having many initial candidates run for the smallest amount of resources or starting with a few candidates immediately benefiting from the maximum resources.

As a result, an execution of Hyperband produces the data $\mathcal{D}^{\text{HB}} = \cup_{b \in \text{brackets}} \mathcal{D}^{\text{HB},b}$, where $\mathcal{D}^{\text{HB},b}$ is formed of all the N_b triplets $\{(\mathbf{x}^n, y^n, r^n)\}_{n=1}^{N_b}$ evaluated in the bracket b, recording the HP candidates, their evaluations and the corresponding resources used.

3 Combining ABLR with Hyperband

We wish to combine ABLR with Hyperband to leverage the attractive properties of the two methods. Next, we describe our approach and argue that it simplifies the methodology developed in [10]. Thanks to the scalability of ABLR, we can maintain a *single* ABLR model *across all resource allocation trade-offs*. The model is learned by ingesting all the data $\mathcal{D}^{\text{HB}}$ produced by Hyperband over the course of the optimization and the brackets (see previous section). Importantly, the resource r is not optimized over since it is fixed by the schedule of Hyperband. It is therefore modeled as a *contextual input variable* (or an environmental variable to reuse the terminology from [9]) of ABLR. More formally, the shared feature map becomes $\boldsymbol{\phi}_{\mathbf{z}}(\mathbf{x}, r) : \mathbb{R}^{P+1} \mapsto \mathbb{R}^D$. This differs from [10], where one TPE model is considered per resource value r, but can span several brackets. This means that no transfer of information is possible across models that use a different number of resources. Moreover, by exploiting all the data generated by Hyperband, we need not introduce an initialization heuristic for the surrogate model when the number of evaluations is too low for some resource value (see Section 4.1 in [10]). Finally, and unlike [10] that introduce an extra multiplicative factor in the bandwidths of TPE, we do not apply any heuristic to adjust the aggressiveness of the exploration.

To query ABLR to suggest a new candidate, we set the contextual variable to $r = r_{\text{current}}$, where r_{current} is the current resource value defined by the Hyperband schedule. We do so for the sake of simplicity, and we leave other options (e.g., setting $r = r_{\max}$) in the spirit of [9], for future work.

Our combination of ABLR with Hyperband straightforwardly extends to the transfer learning setting. Assume we have T tasks with data $\{\mathcal{D}_t^{\text{HB}}\}_{t=1}^T$, corresponding for instance to the application of Hyperband to a given ML model over T different datasets. We adopt the methodology proposed in [20], namely considering one Bayesian linear regression per HP optimization task and sharing the same feature map $\boldsymbol{\phi}_{\mathbf{z}}(\mathbf{x}, r)$ across them. As a result, we obtain a transfer learning variant of Hyperband that can *simultaneously* exploit the data collected during previous Hyperband-based tuning tasks and within the current Hyperband run.

4 Experiments

We consider the following set of competing methods which we divide into two families:

(1) *Resource-unaware* approaches for which one evaluation of f implies using the maximum amount of resources: random search [4] (`random`), standard Gaussian process (`GP`, with the implementation from [32]), ABLR without transfer learning (`ABLR`) and ABLR with transfer learning (`ABLR_transfer`). Here, transfer learning is to be understood as reusing evaluations from previous HP optimization tasks computed with a maximum number of resources.

(2) *Resource-aware* approaches: standard Hyperband (`HB`), the approach of [10] (`HB_BO`, using the code from `https://github.com/automl/HpBandSter`), Hyperband combined with GP (`HB_GP`), Hyperband combined with ABLR without and with transfer learning (respectively `HB_ABLR` and `HB_ABLR_transfer`). `HB_GP` and `HB_ABLR` use the same mechanism to transfer information within a Hyperband: they encode the resource r as a contextual variable (normalized within $[-1, 1]$). `HB_GP` is applicable because the number of evaluations is modest (i.e, only evaluations from within a Hyperband run, which amounts to a total of about several hundreds in our experiments). We did not compare to [9, 13, 14] since [10] was shown to outperform those previous approaches.

Our experimental protocol follows a leave-one-task-out procedure, where a BO problem is solved for a given held-out task using the data from the $T - 1$ remaining tasks. We thereafter report results

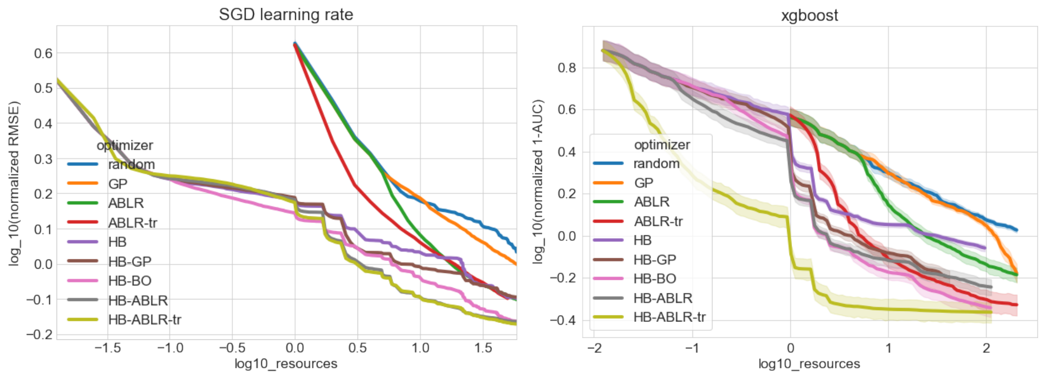

Figure 1: Comparison of different competing methods (left) for the tuning of the learning rate to optimize linear regression problems and (right) for the tuning of XGBoost binary classifiers.

averaged over, both, the T leave-one-task-out folds and the 30 random replications of the experiments. Inspired by [33], we aggregate results across tasks by first normalizing according to the results of random search within each task (`random` has then unit performance, or close to zero if a log transformation is applied as in Figure 1). For `ABLR_transfer` and `HB_ABLR_transfer`, we use transfer learning data resulting from a run of `random` and `HB` respectively (leading in Sections 4.1-4.2 to a total of 206 evaluations per task, associated with either the maximum or various amounts of resources respectively).

4.1 Tuning the learning rate for the stochastic optimization of linear regression models

We consider the problem of tuning the learning rate for the stochastic optimization of linear regression problems. More precisely, denoting the squared loss by $\ell_i(\mathbf{u}) = \frac{1}{2}(\boldsymbol{\theta}_i^\top \mathbf{u} - \tau_i)^2$, we focus on solving

$$\min_{\mathbf{u}\in\mathbb{R}^p} \sum_{i=1}^{n} \ell_i(\mathbf{u}) \text{ with the stochastic gradient (SGD) update rule at step } k\text{: } \begin{cases} \mathbf{v} = \gamma\mathbf{v} + \frac{\nu}{1+\nu\lambda k}\nabla\ell_i(\mathbf{u}^{(k)}) \\ \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \mathbf{v} \end{cases}$$

where $\boldsymbol{\theta}_i \in \mathbb{R}^p$ and $\tau_i \in \mathbb{R}$ refer to input and target of the regression problem, while we use the momentum $\gamma \in [0.3, 0.999]$ together with the learning rate parametrization $\nu \in [0.001, 1.0]$ and $\lambda \in [0.001, 1000.0]$, as advocated in [34]. We tune the HPs $\{\gamma, \nu, \lambda\}$ to optimize the validation RMSE. For the targets, we consider a noisy linear model $\tau_i = \boldsymbol{\theta}_i^\top \mathbf{u}^* + \varepsilon_i$ for some $\mathbf{u}^* \in \mathbb{R}^p$. Using a standard normal distribution, we randomly generate $T = 30$ train/validation problem instances $\mathbf{u}_t^* \in \mathbb{R}^p$, $(\boldsymbol{\Theta}_t^{\text{train}}, \boldsymbol{\varepsilon}_t^{\text{train}}) \in \mathbb{R}^{n\times p} \times \mathbb{R}^n$ and $(\boldsymbol{\Theta}_t^{\text{val}}, \boldsymbol{\varepsilon}_t^{\text{val}}) \in \mathbb{R}^{n\times p} \times \mathbb{R}^n$, with $n = 81$ and $p = 9$. Regarding the resources, we define an unit of resource as three SGD updates and set the maximum amount of resources to 243 (given that $n = 81$, this choice amounts to 3 epochs).

4.2 Tuning XGBoost binary classifiers

In the second experiment, we focus on the tuning of XGBoost binary classifiers to maximize the validation AUC. We tune 8 HPs of XGBoost and consider a subset of $T = 25$ datasets from the `libsvm` repository [35]; the details about the HPs and datasets are provided in the supplementary material. Concerning the resources, we define an unit of resource as one round of XGBoost and set the maximum amount of resources to 81.

4.3 Discussion

The results are reported in Figure 1. On the one hand, we can observe two clusters of curves depending on whether resources are exploited, which turns out to be beneficial at the beginning of the optimization. On the other hand, except for `HB_ABLR` in the SGD setting where performances are comparable, `HB_ABLR_transfer` seems to consistently outperform the other approaches including `HB_BO`. Interestingly, in the XGBoost setting, the gain in performance due to transfer learning already reported in [20] (see `ABLR` and `ABLR_transfer`) appears to carry over as well while using the combination with `HB` (see `HB_ABLR` and `HB_ABLR_transfer`).

References

[1] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. Technical report, preprint arXiv:1603.06560, 2016.

[2] L. Bottou and Y. LeCun. Large scale online learning. In *Advances in Neural Information Processing Systems*, volume 16, pages 217–224, 2004.

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

[4] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[5] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical report, preprint arXiv:1012.2599, 2010.

[6] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[7] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2004–2012, 2013.

[8] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. Technical report, preprint arXiv:1406.3896, 2014.

[9] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. Technical report, preprint arXiv:1605.07079, 2016.

[10] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1436–1445, 2018.

[11] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. Technical report, preprint arXiv:1502.07943, 2015.

[12] Stefan Falkner, Aaron Klein, and Frank Hutter. Combining hyperband and bayesian optimization. In *Proceedings of BayesOpt NIPS workshop*, 2017.

[13] Hadrien Bertrand, Roberto Ardon, Matthieu Perrot, and Isabelle Bloch. Hyperparameter optimization of deep neural networks: Combining hyperband with bayesian model selection. In *Conférence sur l'Apprentissage Automatique (CAP 2017)*, 2017.

[14] Jiazhuo Wang, Jason Xu, and Xuejun Wang. Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning. Technical report, preprint arXiv:1801.01596, 2018.

[15] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 199–207, 2013.

[16] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1077–1085, 2014.

[17] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[18] Matthias Feurer, T Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[19] Nicolo Fusi and Huseyn Melih Elibol. Probabilistic matrix factorization for automated machine learning. Technical report, preprint arXiv:1705.05355, 2017.

[20] Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[21] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *ICML 2018 AutoML Workshop*, July 2018.

[22] Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[23] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4134–4142, 2016.

[24] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl, et al. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554, 2011.

[25] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[26] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

[27] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[28] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of LION-5*, page 507?523, 2011.

[29] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse gaussian processes for Bayesian optimization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

[30] Geoff Pleiss, Jacob R Gardner, Kilian Q Weinberger, and Andrew Gordon Wilson. Constant-time predictive distributions for gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[31] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

[32] `GPyOpt`: A Bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[33] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, 2017.

[34] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[35] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.